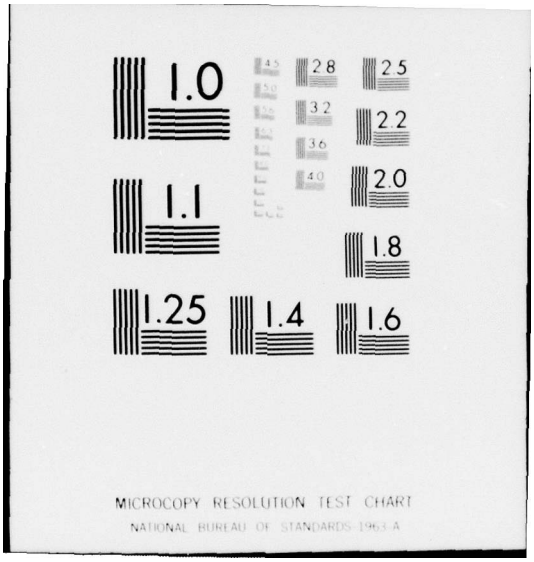
AD-A037 804 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT. UNIVAC SERIES 70145 U--ETC(U)
MAR 77

UNCLASSIFIED CCVS68-VSR185 NL

1 OF 1
ADA037804

END

DATE
FILMED
4-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

FEDERAL COBOL COMPILER TESTING SERVICE

VALIDATION SUMMARY REPORT

Department of the Navy
(ADPESO)

Washington, D.C.
20376

CCVS68-VSR185

COBOL COMPILER
VALIDATION SUMMARY REPORT

VALIDATION NUMBER CCVS68-VSR185

Prepared By:

FEDERAL COBOL COMPILER TESTING SERVICE
DEPARTMENT OF THE NAVY
WASHINGTON, D.C. 20376

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS68-VSR185	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report #CCVS68-VSR185 UNIVAC Series 70/45 USACOB Version 011			5. Report Date 31 March 1977 6.
7. Author(s) Same as organization - see 9.			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Federal COBOL Compiler Testing Service ADP Selection Office Department of the Navy Washington, D. C. 20376			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address ADP Selection Office Department of the Navy Washington, D. C. 20376			13. Type of Report & Period Covered 14.

15. Supplementary Notes

COBOL Compiler Validation Summary Report UNIVAC Series 70/45 USACOB Version 011.

16. Abstracts

This Validation Summary Report (VSR) for the UNIVAC Series 70/45 USA COB COBOL Compiler Version 011 (TDOS Version 23) provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1968 COBOL Standard (X3.23-1968/VIPS PUB 21). The compiler was validated at the High level of FIPS PUB 21. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by level/module within X3.23-1968; a section relating the categories of discrepancies to each of the Federal levels of the language; and a detailed listing of discrepancies together with the tests which were failed.

17. Key Words and Document Analysis. 17a. Descriptors

408 438

Programming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

17b. Identifiers/Open-Ended Terms

CCVS
CVS

17c. COSATI Field/Group 09/02

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited.	UNCLASSIFIED	24
	20. Security Class (This Page) UNCLASSIFIED	22. Price

CCVS68-VSR185

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS68-VSR185
2.	Vendor	Sperry Univac
3.	Mainframe	Series 70 Model 45
4.	Compiler Identification	USACOB Version 011
5.	Operating System Identification	TDOS V-23
6.	CCVS Version	6.2
7.	Federal Information Processing Standard Publication	21

8. For additional information on this compiler contact:

Mr. H. D. Riffel
Sperry Univac
2121 Wisconsin Avenue
Washington, D. C. 20007

PLEASE NOTE: The Federal COBOL Compiler Testing Service may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of this validation are only for the purpose of satisfying United States Government requirements, and apply only to the Computer System, Operating System release, and compiler version identified in the VSR. The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the Federal COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1. Purpose of the Validation Summary Report

The purpose of the Validation Summary Report (VSR) is to identify individual COBOL language elements whose usage does not conform to Federal Standard COBOL as adopted from American National Standard COBOL (X3.23-1968) by Federal Information Processing Standards Publication 21 (FIPS PUB 21).

1.2. Preparation of the VSR

The Validation Summary Report is prepared by analyzing the results of running the COBOL Compiler Validation System (CCVS). The COBOL Compiler Validation System consists of audit routines containing features of Federal Standard COBOL, their related data, and an executive routine (VP-routine) which prepares the audit routines for compilation. Each audit routine is a COBOL program which includes tests and supporting procedures used to indicate the results of the tests.

The testing of a compiler in a particular hardware/operating system environment is accomplished by compiling and executing each audit routine. The report produced by each routine indicates whether the compiler passed or failed the tests contained in the routine.

If the compiler rejects some language elements by terminating compilation, giving fatal diagnostic messages, or terminating execution abnormally, then the test containing the code the compiler was unable to process is deleted and the audit routine compilation and execution repeated.

The compilation listings and the output reports of the audit routines constitute the raw data from which the members of the Federal COBOL Compiler Testing Service produce a Validation Summary Report.

1.3. Organization of the VSR

The Validation Summary Report is made up of several sections the contents of which are described below:

a. Section 2 summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System.

Section 2 is subdivided into Section 2.1, syntax not accepted by the compiler, and Section 2.2, semantic differences between the compiler implementation and the language specification. All errors are grouped by processing module.

b. FIPS PUB 21 defines four levels of Federal Standard COBOL. Section 3 of the VSR lists the discrepancies described in Section 2 by the level in which the problem occurs.

c. Section 4 contains information obtained during the validation process

which, while not impacting the status of the compiler with respect to the defined Federal COBOL levels, may be of interest to a user of the compiler.

d. Section 5 contains information which describes the software environment in which the compiler was tested. This includes the name and version of the compiler; the name and version of the operating system; the implementor-names which were used in the Environment Division of the programs comprising the CCVS; the options used with the compiler; and if applicable, information regarding the use of compiler optimization features.

e. Appendix A is the Validation Summary Working Document, a working paper resulting from the compilation and execution of the CCVS, and from which the VSR is derived.

1.4. Compliance to FIPS PUB 21

Definition of an Implementation of USA Standard COBOL

(excerpts from American National Standard COBOL X3.23-1968, Chapter 1)

Throughout the USA Standard COBOL specifications, there are certain language elements that depend, for their implementation, on particular types of hardware components. The implementor specifies the minimum hardware configuration required for a given implementation of USA Standard COBOL and the hardware components that it supports. Any language elements that depend on hardware components for which support is claimed must be implemented. Language elements that are not implemented because of their dependence on hardware components whose support is not claimed for an implementation must be so specified and their absence will not render the implementation nonstandard.

When a facility is provided that accomplishes the function specified by any particular COBOL element, it may be unnecessary to include that particular element within the COBOL source program. If such an unnecessary element does appear in the source program, it must be accepted by the compiler. However, if the element does not lead to the production of object code, no substitute statements shall be required within the COBOL source program to accomplish this function.

By the same token, the inclusion of language elements or of functions that are not a part of the USA Standard COBOL specifications will not render an implementation of USA Standard COBOL nonstandard. This is true even though it may imply the extension of the list of reserved words by the implementor, and prevent proper compilation of some programs which conform to the Standard.

The Federal COBOL Standard

The Federal COBOL Standard is the same as American National Standard COBOL (x3.23-1968) with two exceptions:

> The Federal Standard defines 4 levels and the ANSI Standard defines only the minimum COBOL implementation and the full standard. Low and High levels of the Federal COBOL Standard (see 1.5) correspond to the above two ANSI levels (minus the Report Writer module). Two additional levels, low-intermediate and high-intermediate have been included in the Federal Standard between the highest and lowest subsets. These additional levels accommodate hardware which cannot support the full standard, but which is capable of implementing more than the minimum standard.
>
> The Report Writer Module has been dropped from the Federal Standard and is not included in the validation process.

The Federal Standard requires that a compiler contain as a minimum the elements specified in at least one of the Federal levels. No restrictions are imposed on the inclusion of selected features from higher levels or even unique vendor extensions. Compatibility among various implementations of a given level containing additonal features must be controlled by management imposed standards and restrictions.

1.5. Federal Standard COBOL Levels

The Federal Government has defined four nested levels of Standard COBOL X3.23-1968. All compilers acquired by a Federal agency must contain as a minimum one of the four Federal levels, depending on machine size, configuration and user needs.

	Low Level	Low-Intermediate Level	High-Intermediate Level	High Level
Nucleus.....	1	2	2	2
FPM				
Table Handling.....	1	2	2	3
Sequential Access....	1	2	2	2
Random Access.....	-	2	2	2
Sort.....	-	-	1	2
Segmentation.....	-	1	1	2
Library.....	-	1	1	2

1.6. Use of the VSR

The Federal COBOL Compiler Testing Service may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation are only for the purpose of satisfying United States Government requirements, and apply only to the computer system, operating system release, and compiler version identified in the VSR.

The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

1.7. Sources of Additional Information

FIPS PUB 21 defines the Federal COBOL Language Standard. This publication is available from the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

The detailed COBOL language specifications are given in the publication "USA Standard COBOL, X3.23-1968", available from American National Standards

Institute, 1430 Broadway, New York, New York 10018.

An explanation of the COBOL Compiler Validation System is contained in the CCVS User's Guide. This document explains how to run the compiler validation system. The User's Guide and a magnetic tape source image copy of the CCVS programs are available from the National Technical Information Service, Springfield, Virginia, 22151. Specify AD772600 for the Users Guide and AD772601 for the CCVS tape.

1.8. Requests for Interpretation

Questions regarding this VSR or the CCVS should be forwarded to the FCCTS. If any problem cannot be adequately resolved through the FCCTS, the request for interpretation will be forwarded to the Federal COBOL Interpretation Committee for final resolution.

A brochure describing the Validation process including the procedures for requesting a validation and resolution of questions involving interpretation of the current Federal Standard is available from the Department of the Navy, Federal COBOL Compiler Testing Service, Washington, D.C. 20376.

1.9. Federal Standard COBOL Approved Interpretation

The National Bureau of Standards published in the Federal Register Vol. 41 No. 179, September 14, 1976, an approved interpretation of Federal Standard COBOL as pertains to the evaluation of arithmetic expressions in the COMPUTE statements. This interpretation states that "size of the intermediate result field is implementor-defined."

Since the results of evaluating arithmetic expressions are not predictable, all COMPUTE statements and IF statements containing arithmetic expressions have been removed from the COBOL Compiler Validation System.

SECTION 2. DETAILED EVALUATION OF ERRORS

This section summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System. The section is subdivided into two sections: Section 2.1. lists the language syntax not accepted by the compiler; Section 2.2. explains the semantic differences between the compiler implementation and the language specification. All errors within each section are grouped by processing module.

Each program in the COBOL Compiler Validation System is identified by a 5-character program name. The name associates the routine with the functional processing module and level of ANS COBOL tested within the program.

The five character name has the general format XXNMM. The first two characters are alphabetic and identify the functional module tested by the program. The permissable values are:

LB - Library
NC - Nucleus
RC - Random Access
SG - Segmentation
SQ - Sequential
ST - Sort
TH - Table Handling

The third character of the audit routine name is either a 1, 2, or 3, and identifies the level of the functional module being tested. Each module and level is represented by several programs. The fourth and fifth characters of the program name are sequence numbers for programs which test features in the same level of the same functional processing module.

As an example, the program name NC210 is the tenth program in the series of routines which test the second level of the Nucleus module.

Each error noted in this section makes reference to a program or functional COBOL module in the COBOL Compiler Validation System.

2.1 Syntactical Errors

2.1.1 Library Module

2.1.1.1 The compiler restricts the library text such that it cannot contain Division headers. This precludes the copying of text containing Division headers and limits inclusion of library text to the division in which the COPY statement resides. (LB106)

2.1.2 Nucleus Module

2.1.2.1 The continuation of non-numeric literals was not interpreted correctly in all cases. (This is a Nucleus level 1 function, LB101.)

2.1.2.2 The continuation of numeric literals and COBOL words was not interpreted correctly in all cases. (This is a Nucleus level 2 function, NC201, NC205.)

2.1.2.3 Multiplying -33 (computational, signed, 18 digit data item) by 99999999999999999 (computational, unsigned, 18 digit data item) results in an answer of:

-265999999999999967

instead of -3299999999999999. (NC101)

2.1.2.4 The figurative constants LOW-VALUE and HIGH-VALUE were rejected as operands in conditional statements which also contained numeric data items as operands. (NC103)

2.1.2.5 Alphanumeric data items whose PICTURE character-string contained combinations of X's, B's, and 0's were treated as numeric elementary data items and, if greater than 18 characters, truncated to 18 positions. (NC105)

2.1.2.6 Procedure-names which consisted solely of numeric characters and were greater than 18 characters in length were not recognized as procedure-names. (NC107)

2.1.2.7 VALUE clauses associated with numeric edited data items were rejected. The VALUE clauses contained an alphanumeric literal and the figurative constant ZERO. The diagnostics issued by the compiler suggested that there was a class conflict between the PICTURE character-string and the value clause. (NC107, NC108, NC206)

2.1.2.8 A CURRENCY SIGN clause was not interpreted correctly. The substitute currency symbol was specified as being the equal sign (=). PICTURE character-strings which

contained the substitute currency symbol were rejected as being syntactically incorrect. (NC108)

2.1.2.9 STOP statements with signed numeric literals print the last digit as an alphanumeric character. The literal -7.1 prints as R7J and +123456789987654321 prints as 1234567899876543 2A. (NC109)

2.1.2.10 A DISPLAY statement mixing literals, alphanumeric data items and numeric data items was flagged as incorrect with a message stating there were conflicting classes of operands in a DISPLAY statement. (NC109, NC204)

2.1.2.11 Comparisons (relation conditions) between elementary numeric data items and the figurative constants QUOTE(S) and HIGH-VALUE(S) were rejected. (NC201)

2.1.2.12 A paragraph in the Procedure Division of one of the programs was rejected because it was 'too long' (i.e., too many statements under a given procedure-name). The paragraph had to be broken up into several paragraphs in order to compile the source program. (NC202)

2.1.2.13 The RENAMES clause

```
66  dn1 RENAMES dn2 THRU dn3 IN dn4
```

was flagged as an invalid clause. The program compiled correctly when the IN was changed to OF. (NC203)

2.1.2.14 A MOVE CORR statement with no corresponding items was correctly flagged as having no corresponding items, but this was considered a serious syntactical error (severity code = 2). (NC209)

2.1.2.15 The statement

```
IF identifier EQUAL TO SPACE
```

where identifier is defined as PICTURE *999999 was rejected by the compiler. (NC212)

2.1.3 Random Access Module

The Random Access Module is not implemented according to the specifications in X3.23-1968. If the ACCESS MODE IS RANDOM for a file, only OPEN INPUT or OPEN I-O may be executed for the file. The OPEN OUTPUT statement is not permitted.

2.1.4 Sequential Access Module

2.1.4.1 All user defined file-names must be unique within the

first seven characters of the file-name. All programs were modified as necessary to accommodate this limitation.

2.1.4.2 The word THROUGH was rejected as being syntactically inappropriate in the FILE-LIMITS clause of the SELECT statement. (SQ103)

2.1.4.3 The Sequential Access Module level 2 programs were not run for this validation since Declarative USE procedures are not supported according to the specifications in X3.23-1968.

2.1.5 Sort Module

2.1.5.1 An elementary numeric data item 18 digits in length was rejected as a sort key because it was greater than 16 digits in length. (ST106)

2.1.5.2 The Sort Module level 2 routines were not run for this validation.

2.1.6 Table Handling Module

2.1.6.1 Any SEARCH statement containing the VARYING identifier phrase was rejected with a compiler message stating there was an invalid operand. (TH301)

2.1.6.2 The OCCURS DEPENDING clause is not handled correctly. An "OCCURS 0 TO 26 DEPENDING identifier" was flagged during compilation with a message stating 0 was greater than 26. (TH308)

2.2 Semantic Errors

2.2.1 Nucleus Module

2.2.1.1 Dividing -33 (computational, signed, 18 digit data item) into 66666666666666666 (computational, unsigned, 18 digit data item) resulted in an answer of:

-020202020202020195

instead of -020202020202020202. (NC101)

2.2.1.2 The actual size of an elementary numeric data item whose USAGE IS COMPUTATIONAL is not determined by the length specified in the PICTURE character-string associated with the data item. As a result, truncation of the receiving data item and size error conditions for the resultant identifier in a arithmetic operation did not occur as was expected. (NC105)

2.2.1.3 Moving an alphanumeric literal containing only numeric characters to an unsigned elementary numeric item whose USAGE IS DISPLAY did not result in the expected left truncation of the sending item. (NC105)

2.2.1.4 When moving the figurative constant ZERO to an alphanumeric edited data item only a single character zero is moved, not a string of zeros equal to the length of the receiving data item. (NC105)

2.2.1.5 The special register TALLY, for this implementation, is signed (should be unsigned) and is greater than 5 digits in length (should be exactly 5 digits in length). (NC111)

2.2.1.6 An EXAMINE statement containing a subscripted data-name failed to execute correctly. The item to be examined is defined as PICTURE X and is the first of three occurrences. The EXAMINE statement contains the phrase

REPLACING LEADING "Z" BY ZERO.

The single "Z" was not replaced by a zero. (NC111, NC206)

2.2.1.7 A PERFORM statement (option 4) with a FROM phrase containing a negative numeric literal did not execute correctly. (NC201)

2.2.1.8 An abbreviated combined relation condition in an IF statement was not evaluated correctly. The IF statement and the values for each variable are:

```
IF A LESS THAN B AND NOT GREATER THAN (C OR D)
    ... ELSE ...
```

with A = 5, B = 7, C = 1, and D = 6. The ELSE path was not executed for this statement. (NC201)

2.2.1.9 A MOVE of the numeric literal 234.5 to a data item which RENAMES an elementary item with PICTURE ***,***.** failed to execute correctly. The result after execution of the MOVE statement was equal to spaces (the initial value) instead of the expected results of ****234.50. (NC203)

2.2.2 Sequential Access Module

2.2.2.1 The first character of each record written to the printer was not printed. (SQ101)

2.2.2.2 The action taken, when using the ADVANCING phrase of

the WRITE statement for printer destined files, is incorrect when the number of lines being requested to advance is specified as zero. In this case, instead of no paper movement, a page eject is issued. (SQ101)

2.2.2.3 No restart tape was created even though the RERUN clause

RERUN ON SYS033 EVERY 10 RECORDS OF FILE1

was included in the I-O-CONTROL paragraph. (SQ106)

2.2.3 Sort Module

2.2.3.1 Sort keys which were defined as signed elementary numeric data items whose USAGE was COMPUTATIONAL were not processed correctly during the sorting operation. Negative numbers in the data for that field were not considered to be less than positive numbers. (ST108)

2.2.4 Table Handling

2.2.4.1 A SEARCH of a table which was defined with the OCCURS DEPENDING clause does not execute correctly if the table length is zero. (TH308)

SECTION 3. COMPILER STATUS

Section 1.5 explains the four levels of Federal Standard COBOL. This section lists the discrepancies described in Section 2 by the Federal level in which the problem occurs. All errors listed for a lower level are also errors in any higher level, even though they are listed only in the lower level. The paragraph number from Section 2 is used to reference the errors in each Federal level.

3.1 Low Level

3.1.1 NUCLEUS LEVEL 1

2.1.2.1 Continuation of non-numeric literals.
2.1.2.3 Multiplying computational data items 18 digits in length.
2.1.2.4 Comparison of numeric data item to LOW-VALUE/HIGH-VALUE.
2.1.2.5 PICTURE character-string (X, B, 0) with numeric data item characters.
2.1.2.6 Procedure names, numeric and greater than 18 characters.
2.1.2.7 Value ZERO and alphanumeric literal with numeric item.
2.1.2.8 Currency symbol substitution.
2.1.2.9 STOP statement with signed numeric literal.
2.1.2.10 DISPLAY statement mixing alphanumeric and numeric identifiers and literals.
2.1.2.11 Comparison of QUOTE/HIGH-VALUE to numeric data item.
2.1.2.12 Paragraph too big.
2.1.2.15 IF id (PIC *999999) EQUAL TO SPACE rejected.
2.2.1.1 Dividing computational data items 18 digits in length.
2.2.1.2 Size of computational items differs from PICTURE.
2.2.1.3 Moving alphanumeric literal to numeric data item.
2.2.1.4 Moving ZERO to alphanumeric edited data item.
2.2.1.5 TALLY greater than 5 digits and signed.
2.2.1.6 EXAMINE statement with qualified data-name, subscripted, and length one.

3.1.2 SEQUENTIAL ACCESS LEVEL 1

2.1.4.1 Filenames are limited to 7 characters.
2.1.4.2 THROUGH is not equivalent to THRU.
2.2.2.1 First character of print line is truncated.
2.2.2.2 ADVANCING zero lines causes page eject.
2.2.2.3 No restart file created for RERUN clause.

3.2 Low-Intermediate Level

3.2.1 NUCLEUS LEVEL 2

2.1.2.2 Continuation of numeric literals and COBOL words.
2.1.2.13 RENAMES clause containing qualifier IN.
2.1.2.14 Rejected MOVE CORR statement with no corresponding item.
2.2.1.7 PERFORM ... VARYING ... FROM negative numeric literal.

2.2.1.8 Abbreviated combined relation condition.
2.2.1.9 MOVE to item which RENAMES elementary item with PICTURE ***,***.**.

3.2.2 SEQUENTIAL ACCESS LEVEL 2

2.1.4.3 Declarative (USE) procedures.

3.2.3 RANDOM ACCESS LEVEL 2

2.1.3 Random Access not implemented according to X3.23-1968 .

3.2.4 LIBRARY LEVEL 1

2.1.1.1 Division header is not permitted in library text.

3.3 High-Intermediate

3.3.1 SORT LEVEL 1

2.1.5.1 Numeric sort key is limited to 16 characters.
2.2.3.1 Signed computational data sorts incorrectly.

3.4 High Level

3.4.1 TABLE HANDLING LEVEL 3

2.1.6.1 SEARCH ... VARYING identifier statement.
2.1.6.2 OCCURS 0 TO integer-2 DEPENDING clause.
2.2.4.1 SEARCH of table whose variable length is zero.

SECTION 4. INFORMATION ITEMS

Section 4.1 covers the cases where the results of a given statement are not defined in the language specifications.

Section 4.2. gives other information about the compiler which was discovered during validation. Included in this section are items which are hardware dependent, and situations where implementor defined variations are permitted.

Section 4.3. gives information on hardware dependent features that are not supported by the subject COBOL Compiler.

Section 4.4 gives information concerning the use of facilities outside the COBOL program that accomplish functions defined in COBOL (see 1.4 of this VSR).

The COBOL language specification "American National Standard COBOL X3.23-1968" as defined by FIPS Pub 21 is used as the reference document.

4.1. Information Tests

4.1.1. Nucleus Module

4.1.1.1. CLOSE followed by an *OPEN OUTPUT of a printer destined file* (LB103):

COPY-TEST-2 of LB103 closed and then reopened the printer file. The result of closing and then reopening a unit record output file is not specifically addressed in the language specifications. The object of this test is to determine whether the file is repositioned to its beginning and all previous results overwritten, or whether all current output is preserved with additional output concatenated to it.

The results for this compiler:

All printed output was provided.

4.1.1.2. IF...Signed Numeric item equals Alphanumeric item (NC103):

IF-TEST-65 compares two elementary items with the first item having PIC S9(18) VALUE 111111111111111111 and the second item having a PIC X(18) VALUE "111111111111111111". The items are compared by the form:

IF identifer-1 EQUAL TO identifier-2...

The results for this compiler:

The two elementary items did compare equal.

4.1.1.3. IF...NUMERIC Condition tests (NC103):

CLASS-TEST-17 and CLASS-TEST-22 of NC103 are information tests where data items with PICTURE S9 contain -1 and +1 respectively. Each data item is redefined

as an alphanumeric data item PICTURE X. The redefined data item is then referenced in a NUMERIC test to determine whether the signed data item contains an indicator to represent the sign, i.e. an overpunch. If both data items are determined to be numeric, then the system does not use an indicator technique for the sign. If both data items are determined to be not numeric then the system uses an indicator method for both positive and negative numbers. If one of the data items is numeric and the other is not, then the system uses the indicator method for one case but not the other. The sign is not addressed by the COBOL language specifications and consequently is left to the discretion of the implementors of COBOL compilers as to its representation.

The results for this compiler:

Negative data redefined as unsigned data:

The data item tested NOT NUMERIC.

Positive data redefined as unsigned data:

The data item tested NOT NUMERIC.

4.1.1.4 Move signed numeric field to alphanumeric field (NC105):

MOVE-TEST-148 and MOVE-TEST-149 in NC105 move signed numeric fields with PICTURE S9(5) to fields with PICTURE X(5). The source fields had contents of +60666 and -70717 for MOVE-TESTS-148 and 149 respectively. The language specification is not definitive as to whether the absolute value is to be moved (i.e. the sign is stripped) or whether the contents are moved without regard to the sign. The results of these tests should be considered only in light of the results of the information test in 4.1.1.2.

The results for this compiler:

MOVE +60666 TO X(5)

The absolute value was moved; the sign was stripped.

MOVE -70717 TO X(5)

The absolute value was moved; the sign was stripped.

4.1.1.5. ADD without ON SIZE ERROR (NC106):

ADD-TEST-17 in NC106 checks the effect of a size error on the result of an ADD statement which does not have the ON SIZE ERROR phrase. The statement used was adding SV9(5), S9(6)V9(6), SV9(5) GIVING S9(5) ROUNDED. Identifier S9(6)V9(6) contained the value 333333.333333. The SV9(5) descriptions were the same identifier and contained the value .11111. The language specifications do not define the results of an arithmetic operation without the SIZE ERROR phrase, when the result cannot be contained in the resultant-identifier which causes truncation of significant digits. The results of this test require 6 digits and the resultant-identifier contains only 5. For a further discus-

sion, see X3.23-1968 American National Standard COBOL page 2-28, Section 6.2.2(1), The SIZE ERROR clause.

The results for this compiler:

The result was stored in the resultant-identifier with the high order digit truncated.

4.1.1.6. Abbreviated Combined Relation Conditions (NC201):

IF--TEST-110 in NC201 utilizes an abbreviated combined relation condition to determine the direction the compiler will take in a case where the use of the word NOT is indeed confusing. The language specification states on page 2-73, section 5.2.1.1 that when the construct AND NOT LESS THAN is used in a source program, the word NOT is to be treated as a logical operator. This becomes relevant when taken in the context of the later abbreviation of the relational operator in the statement. For example, A = B AND NOT LESS THAN C OR D. would expand to the following: A = B AND NOT A LESS THAN C OR A LESS THAN D. The test in question contains the following: A GREATER THAN B AND IS NOT LESS THAN C OR D which strongly suggests that the word NOT in this case is relational in nature and not logical. The concept of an optional word (IS) changing the meaning of a statement is contrary to the philosophy of COBOL and as a result this test is presented as information only.

The values for the variables in the condition are A=5, B=7, C=1 and D=6. Using these values in place of the variables, the condition expands as the following if NOT is treated as relational:

((5 GREATER THAN 7) AND (5 NOT LESS THAN 1)) OR (5 NOT LESS THAN 6)

In this case the condition is false.

If the NOT is logical, the condition expands as the following:

((5 GREATER THAN 7 AND NOT (5 LESS THAN 1)) OR (5 LESS THAN 6)

In this case the condition is true.

The results for this compiler:

The word NOT was treated as a relational operator.

4.1.1.7 The Synchronized Clause (NC107)

The language specification does not specify the results of the use of the synchronized clause without the specification of either LEFT or RIGHT. In the event that such results are provided during the validation process, they are included here as information.

The results for this compiler:

The compiler did not indicate how the synchronization would be done.

4.1.2. Random Access Module

4.1.2.1 Multiple length records (RC102):

Program RC102 produces a file containing records of multiple sizes. The purpose of this test is to determine whether multiple size records are written or whether the maximum size record is always written. As the file is created, records of both length are written; and as the smaller records are built, information uniquely identifying each record is placed in the portion of the larger record that would not logically be written to the external media. When the file is later retrieved the record area is examined to determine whether the extra portion beyond the end of the smaller record is available when each of the smaller records is read. The language specificaion does not readily suggest that the external media will be impacted by the size of the logical record as defined in the COBOL program.

The results for this compiler are:

The compiler does not support the Random Access Module.

4.1.3. Sequential Access Module

4.1.3.1. Multiple Length Records for Tape Files (SQ102):

The audit routine SQ102 creates a tape file with multiple length records. The file is then read and the record area examined to determine whether fixed length records were written. For a discussion of this operation, refer to 4.1.2.1. under the discussion of Random Access.

The results for this compiler are:

The compiler appears to support multiple length records.

4.1.3.2. Multiple length records for sequential mass-storage files (SQ104):

The audit routine SQ104 creates a sequential mass-storage file with multiple length records. The file is then read and the records are examined to determine whether fixed length records are written. Refer to 4.1.2.1 under the discussion of multiple records for Random Access Files for a description of a similar test.

The results for this compiler are:

The compiler appears to support multiple length records.

4.1.3.3. FILE-LIMITS Clause

The FILE-LIMITS clause need not be used by the compiler and/or operating system to allocate file space. The language specification allows an external source or function to accomplish some functions described in the COBOL language, in particular, file facilities management functions.

The results for this compiler are:

The FILE-LIMITS clause is treated as comments by this compiler in that file allocation is handled through control cards.

4.1.3.5 RERUN Clause Option

The language specification requires that the implementor must provide at least one of the specified forms of the RERUN clause.

The RERUN clause used for this compiler was:

    RERUN ON SYSC33 EVERY 10 RECORDS OF file-name.

4.2 Other Information

4.2.1. Library Source Text

The language specification is somewhat vague as to the content of the library text prior to being copied. As a result there are a multitude of techniques that have been noted for establishing the source text for subsequent use by a COPY statement. This information is provided in order to understand the technique used by this system.

The results for this compiler are:

There were no modifications required for the library text.

4.3 Hardware dependent features.

o Hardware Switches

This system does not support hardware switches and as a result, all tests related to the testing of switches were deleted.

o OPEN REVERSED Option

This system does not support the use of REVERSED on non mass-storage devices. As a result, all tests related to testing of the REVERSED option were deleted.

4.4 Transparent Implementation

The VALUE OF clause of the File Description Entry.

The FILE-LIMITS clause of the SELECT statement.

SECTION 5 SOFTWARE ENVIRONMENT.

The compiler referenced in this document was validated using the software environment described in this section. When using a modification of the described environment, the compiler may or may not continue to conform to the standard. It should be noted that during the validation process, an attempt is made to validate as many different options as possible.

The use of compiler options, implementor-names in the Environment Division and any form of optimization which is not described in this report could cause the compiler to produce a program that does not perform according to the specifications of Standard COBOL. Only the environment described in this document has been used with this compiler to satisfy the requirements of FIPS PUB 21 and FPMR 101-32.1305.1a. (Any deviations which must be corrected as per the referenced FPMR are described in Sections 2 and 3 of this report.)

1. Options or parameters used on the processor call statement for the compiler: The following options/parameters were used during the validation.

Options specified:

CODE=3,LIST=YES

2. Environment Division implementor-names.

Printer destined files

SYSLST RESERVE NO

Tape files

UT-442-S-SYSnnn

Sequential Mass-storage files

UT-564-S-SYSnnn

Random Access files

Not supported by the compiler.

Sort files (SD)

DISC

Switch names

Not supported by the compiler.

Source Computer names

UNIVAC-SERIES-70 70-45

Object Computer names

UNIVAC-SERIES-70 70-45 MEMORY SIZE 524288 CHARACTERS

3. Optimization. The compiler may or may not have optimization features. If there was an optimization feature available, it was used during the validation process (during a separate execution of the Compiler Validation System) to determine if its use causes the compiler to produce a program which does not give the expected results. If the optimization is invoked through the compiler call statement then it is mentioned in paragraph 1 above. If it is invoked through the introduction of syntax in other than the Data and Procedure Divisions of the source program it is shown below. Optimization which would require modification to the Data and Procedure Divisions is not considered in this report in that it is beyond the scope of the use of standard COBOL and the validation process.

No optimization was used during the validation of this compiler.

4. Compiler.

USACOB Version 011 released 28 June 1974.

5. Operating system.

TDOS V-23.